PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 :
H04L 9/30

A1

(11) International Publication Number: WO 00/05837

(43) International Publication Date: 3 February 2000 (03.02.00)

(21) International Application Number: PCT/CA99/00658

(22) International Filing Date: 21 July 1999 (21.07.99)

(30) Priority Data:
2,243,761 21 July 1998 (21.07.98) CA

(71) Applicant *(for all designated States except US)*: CERTICOM CORP. [CA/CA]; Suite 103, 200 Matheson Bouleward West, Mississauga, Ontario L5R 3L7 (CA).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: VADEKAR, Ashok [CA/CA]; 250 Harris Street, R.R. 4, Rockwood, Ontario N0B 2K0 (CA). LAMBERT, Robert, J. [CA/CA]; 63 Holm Street, Cambridge, Ontario N3C 3N3 (CA).

(74) Agents: PILLAY, Kevin et al.; Orange Chari Pillay, Toronto Dominion Bank Tower, Toronto–Dominion Centre, Suite 3600, P.O. Box 190, Toronto, Ontario M5K 1H6 (CA).

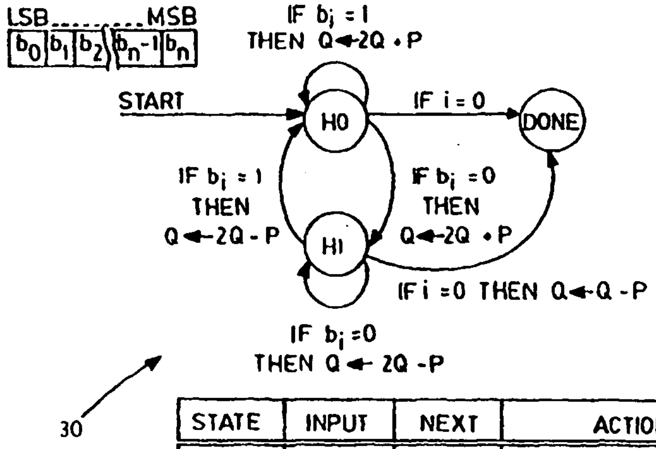
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: TIMING ATTACK RESISTANT CRYPTOGRAPHIC SYSTEM

(57) Abstract

A method for determining a result of a group operation performed an integral number of times on a selected element of the group, the method comprises the steps of: representing the integral number as a binary vector; initializing an intermediate element to the group identity element; selecting successive bits, beginning with a left most bit, of the vector. For each of the selected bits; performing the group operation on the intermediate element to derive a new intermediate element; replacing the intermediate element with the new intermediate element; performing the group operation on the intermediate element and an element, selected from the group consisting of: the group element if the selected bit is a one; and an inverse element of the group element if the selected bit is a zero; replacing the intermediate element with the new intermediate element. In a final step, performing the group operation on the intermediate value and the inverse element if the last selected bit is a zero; and replacing the intermediate element therewith, to obtain the result, whereby each of the bits of the integral is processed with substantially equal operations thereby minimizing timing attacks on the cryptographic system.


LSB.........MSB
b0 b1 b2 bn-1 bn
IF bi = 1 THEN Q ← 2Q + P
START
H0
IF i = 0
DONE
IF bi = 1 THEN Q ← 2Q - P
IF bi = 0 THEN Q ← 2Q + P
H1
IF i = 0 THEN Q ← Q - P
IF bi = 0 THEN Q ← 2Q - P
30

STATE	INPUT	NEXT	ACTION
H0	i < 0	DONE	SUBTRACT
H0	b_i = 0	H1	DOUBLE, ADD
H0	b_i = 1	H0	DOUBLE, ADD
H1	i < 1	DONE	
H1	b_i = 0	H1	DOUBLE, SUBTRACT
H1	b_i = 1	H0	DOUBLE, SUBTRACT

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

TIMING ATTACK RESISTANT CRYPTOGRAPHIC SYSTEM

The present invention relates to the field of cryptographic systems and in particular to a method and apparatus for resisting timing attacks on a cryptographic system.

BACKGROUND OF THE INVENTION

Cryptographic systems generally owe their security to the fact that a particular piece of information is kept secret without which it is almost impossible to break the scheme. This secret information must generally be stored within a secure boundary, making it difficult for an attacker to get at it directly. However, various schemes or attacks have been attempted in order to obtain this secret information. One of these is the timing attack.

By way of background current public key cryptographic schemes such as RSA and elliptic curve (EC) operate over mathematical groups F_p^* and $E(Fq)$ respectively. The group operations, called multiplication modulo p, in RSA, and addition of points in EC are repeated in a particular way to perform a scalar operation. In RSA the operand is called an exponent, the operation is called exponentiation and the method of multiplying is commonly known as repeated square-and-multiply. Thus given a number a $\in F_p$ and an integer $0 \leq k < p$, the exponent, whose binary representation is $k = \Sigma_{i=0}^{t} k_i 2^i$ a value a^k mod p may be calculated by repeated use of the square-and-multiply algorithm. Similarly given $g(x) \in F_p m$ and an integer $0 \leq k \leq p^m - 1$ then $g(x)^k$ mod $f(x)$ may be calculated by this method.

On the other hand, in EC the operand is a scalar multiplier, the operation is called scalar multiplication of a point, and the method is known as double-and-add. Thus if a is a positive integer and P is an elliptic curve point then aP may be obtained by the double-and-add method. Both these methods are well known in the art and will not be discussed further.

In RSA, half of all exponentiation operations use a private key. Whereas in EC all scalar multiplications use either a long term private key or a session private key. In each of these cases, the private key is safe due to the difficulty of reversing the exponentiation or multiplication operation as the case may be. This is based on the discrete log problem or the difficulty of integer factorization. As mentioned earlier, an attacker once in possession of the private key (either long term or session) is able to forge signatures and decrypt secret messages for the attacked entity. Thus it is paramount to maintain the secrecy or integrity of the private key in the system.

Many techniques have been suggested to obtain the private key. The encryption operations are performed either in a special purpose or general purpose processor operating in a cyclic manner. Recent attack methods proposed in open literature have been based on timing analysis of these processors or in other words timing analysis of 'black box' operations. In one instance an attacker by capturing the instantaneous power usage of a processor throughout a private key operation obtains a power signature. The power signature relates to the number of gates operating at each clock cycle. Each fundamental operation as described in the preceding paragraph generates a distinct timing pattern. Other methods exist for obtaining a power signature than instantaneous power usage.

Laborious but careful analysis of an end-to-end waveform can decompose the order of add-and-double or square-and-multiply operations. Either a double or square must occur for each bit of either the exponent or scalar multiplier respectively. Therefore, the places where double waveforms are adjacent each other represent bit positions with zeros and places where there are add patterns indicate bits with ones. Thus these timing measurements can be analyzed to find the entire secret key and thus compromise the system. Thus there is a need for a system which minimizes the risk of a successful timing attack.

SUMMARY OF THE INVENTION

This invention thus seeks to provide a cryptographic system where cryptographic operations are performed by a processor in a constant period of time irrespective of the operation being performed whereby a constant amount of time is required for the processing of each bit scalar or a exponent regardless of its value.

A method for determining a result of a group operation performed an integral number of times on a selected element of the group, said method comprising the steps of :

(a) representing said integral number as a binary vector;

(b) initializing an intermediate element to the group identity element;

(c) selecting successive bits, beginning with a left most bit, of said vector and for each of said selected bits;

(i) performing said group operation on said intermediate element to derive a new intermediate element;

(ii) replacing said intermediate element with said new intermediate element;

(iii) performing said group operation on said intermediate element and an element, selected from the group consisting of:

said group element if said selected bit is a one; and

an inverse element of said group element if said selected bit is a zero;

(iv) replacing said intermediate element with said new intermediate element;

(d) performing said group operation on said intermediate value and said inverse element if said last selected bit is a zero; and replacing said intermediate element therewith, to obtain said result, whereby each of the bits of said integral is processed with substantially equal operations thereby minimizing timing attacks on said cryptographic system.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features of the invention will now be described by way of example only with reference the accompanying drawings in which:

Figure 1 is a schematic representation of a data communication system;

Figure 2 is a schematic representation of a binary number recoding scheme;

Figure 3 is a state machine representation for compiling a scalar multiple of a point;

Figure 4 is a pseudocode implementation of the state machine of Figure 3;

Figure 5 is a non-state machine pseudocode implementation;

Figures 6 and 7 are respectively a pseudocode and state-machine implementation for a square and multiply scheme; and

Figure 8 is a generalized schematic diagram of an apparatus for implementing the method according to an embodiment of the invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Referring therefore to Figure 1, a secure data communication system 10 includes a pair of correspondents, designated as a sender A(12), and a recipient B(14), who are connected by a communication channel 16. Each of the correspondents A and B (12,14) includes an encryption unit 18,20 respectively that may process digital information and prepare it for transmission across the channel 16.

Generally, the sender A assembles a data string, which includes amongst others the public key y of the sender, a message m, the sender's short-term public key k and signature S of the sender A. When assembled the data string may be forwarded to the intended recipient B, who then verifies the signature using A's public key. This public key information may be obtained from a certification authority (CA) 24 or sometimes is sent with the message.

For example, in RSA public key encryption, B encrypts a message m for A, which A decrypts. To encrypt the message m, B obtains A's public key (n, e) and represents the message as an integer in the interval [0, m-1]. Next B computes $c = m^e \bmod n$ and sends the cipher text c to A. The entity A then recovers the message m from c by using its private key d to recover $m = c^d \bmod n$. The calculation $c^d \bmod n$ (modulo exponentiation) may be performed by using a well known square and multiply algorithm. Similar computations are required for signing in RSA, and for signing, encryption and decryption in discrete log systems such as ECC.

The values used in the computations are expressed as bit vectors, which are then manipulated by the encryption processor in accordance with a particular encryption scheme being used. Thus, referring to figure 2, any bit vector ending in a one can be recoded into an all non-zero vector of +1 and -1 terms. For elements ending in a zero, it is necessary to either maintain the final zero in the recoded form or to perform a further -1 action as a correction after the final zero is processed by the previous rules. As will be apparent later, it is preferable to do the corrective action.

Referring to figure 3 a state machine implementation of an algorithm for computing an integer multiple b of a point P on an elliptic curve is shown generally by numeral 30. In this embodiment, a bit vector b to be processed is fed into the state machine 30. The result of the processing is the value bP where P is a point having coordinates (x,y) on an elliptic curve. The state machine 30 is initialized with a counter *i* set to the number of bits N in the vector b, and an intermediate value Q stored in a register. Entry into state H0 of the state machine occurs when the first bit (MSB) of b is encountered (not the first non-zero bit). In state H0 the contents of the intermediate Q is doubled and the base point P is added thereto. The bits are sequentially processed causing transitions either back to the current state H0 or to a next state H1. In state H1, Q is doubled and the base point P is subtracted. When the next bit is a zero, the next state is always H1. When the next bit is a one, the next state is always H0. Whenever the current state is H0, an add (+P) occurs, otherwise a subtract (-P) occurs. Once all bits are processed, control exits to the DONE state with the result bP. If the exit condition is encountered while in H0, it is necessary to subtract (-P) a final time from the result.

Implementation of the state machine in pseudocode form requires the availability of a general purpose storage bit for the H state. Figure 4 shows a pseudocode implementation of the state machine where general purpose data and control mechanisms are available. Timing attacks are prevented only if the execution time and power are identical for all possible execution paths through the loop. In this implementation, this requires that the time for the execution path through lines L5, L6 and L7 be the same as that through lines L5, L8 and L9. This implies that the branch instruction (IF) must execute in the same time for false and true conditionals.

Furthermore, the add and subtract operations must take the same time. By adding in the negative value of P, this is more obviously possible. Note that the execution time of all other lines in the algorithm are non-critical to the timing attack resistance, with the exception of lines L14 and L15. These two lines, if executed will increase the total time required, thus revealing that the final H state was zero. This is equivalent to revealing that the final bit of the scalar was a zero. This is unavoidable, but only reveals a single bit of the scalar multiple.

On application specific computing devices, it is likely that there are no general purpose data storage areas nor general purpose assignment and test operators. In this case, the H state control cannot be added in a usual way. Instead, it is necessary to encode the state by branching through distinct code paths.

Referring to Figure 5 shows a 'state-less' pseudocode implementation. Each code execution path corresponds to a distinct state. While this will in general cause code expansion, here there are only two short paths required. Here, as above, timing attacks are prevented only if the execution time and power are identical for all possible execution paths through the loop. In this implementation, this requires that the time for the execution path through lines LL3, LL4 and LL5 be the same as that through lines LL6, LL7 and LL8. In addition, path LL9, LL10 and LL16 must execute in the same time and power as path LL9 and LL11. This will be true in architectures where conditional branch instructions take the same time to branch (to a new location) as to fall through (to the following location). Note that lines LL11, LL12 and LL13 are equal in execution time and power to lines LL16, LL17 and LL18. This is also necessary to prevent timing attacks. Otherwise, H state information would be revealed. As in the previous implementation, a final corrective subtract is required line LL14 when the loop terminates through the H0 path. This again reveals the final H path, or equivalently the final bit of the scalar.

Referring to Figure 6 a pseudocode implementation of a square and multiply operation is shown by numeral 60, while a corresponding state machine implementation is shown in Figure 7. In this implementation, the exponent bit vector b is fed into the state machine 60. The result of the processing is to compute a value M^b. Once again a counter is initialized to the length N of vector b and an accumulator Q is initialized to one. Entry into state H0 of the state machine begins with the MSB of bit vector b. As previously, the bits are sequentially processed causing transitions either back to the current state H0 or to a next state H1. In state H0 the contents of the accumulator Q is squared and multiplied by the base M. In state H1 the accumulator is also squared and then divided by the base M. If the next bit is a zero, the next state is always H1, whereas if the next bit is a one the next state is always H0. Whenever the current state is H0, a multiply occurs otherwise a divide occurs. As previously, once all bits are consumed, controls exits to the DONE state. If the exit condition is encountered while in H0, it is necessary to divide a final line by the base M.

Figure 8 shows a generalized processor implementation in which a state controller 82 is programmed to run the estate code as described earlier. A modulo arithmetic and/or finite field computation unit 84 is provided for computing the square and multiplication and the point additions/subtractions respectively. A counter 86 and a register b is coupled to the state controller for timing the sequential operation of the controller 82.

While the invention has been described in connection with a specific embodiment thereof and in a specific use, various modifications thereof will occur to those skilled in the art without departing from the spirit of the invention as set forth in the appended claims.

The terms and expressions which have been employed in the specification are used as terms of description and not of limitations, there is no intention in the use of such terms and expressions to exclude any equivalents of the features shown and described or portions thereof, but it is recognized that various modifications are possible within the scope of the claims to the invention.

THE EMBODIMENTS OF THE INVENTION IN WHICH AN EXCLUSIVE PROPERTY OR PRIVILEGE IS CLAIMED ARE DEFINED AS FOLLOWS:

1. A method for determining a result of a group operation performed an integral number of times on a selected element of the group, said method comprising the steps of :

(a) representing said integral number as a binary vector;

(b) initializing an intermediate element to the group identity element;

(c) selecting successive bits, beginning with a left most bit, of said vector and for each of said selected bits;

(i) performing said group operation on said intermediate element to derive a new intermediate element;

(ii) replacing said intermediate element with said new intermediate element;

(iii) performing said group operation on said intermediate element and an element, selected from the group consisting of:

said group element if said selected bit is a one; and

an inverse element of said group element if said selected bit is a zero;

(iv) replacing said intermediate element with said new intermediate element;

(d) performing said group operation on said intermediate value and said inverse element if said last selected bit is a zero; and replacing said intermediate element therewith, to obtain said result, whereby each of the bits of said integral is processed with substantially equal operations thereby minimizing timing attacks on said cryptographic system.

2. A method as defined in claim 1, said group being a multiplicative group F_p* said group element being an integer, and said group operation being exponentiation g^a and an inverse element being the multiplicative inverse 1/g.

3. A method as defined in claim 1, said group being an additive group E(F_{2^M}) and said group operation being addition of points.

4. A method as defined in claim 1, said group being an additive group E(F_q), said group element being a point P with coordinates (x,y) on the elliptic curve, and said group operation being the scalar multiple kP of said point and an inverse element being the negative –P of said point.

5. A method as defined in claim 1, said integral value being a private key k.

6. A method of performing a selected group operation on a scalar and a selected element of said group, in a cryptographic processor, said method comprising the steps of :

representing said scalar as a binary vector;

recoding said binary vector to produce a signed digit representation of plus one and minus one digits;

selecting each of said recoded bits sequentially and for each of said selected bits performing said group operation on an intermediate element to derive a new intermediate element; and adding or subtracting said selected element to said intermediate element in accordance with said sign if said digit being selected; and

outputting said intermediate value as a result of said group operation.

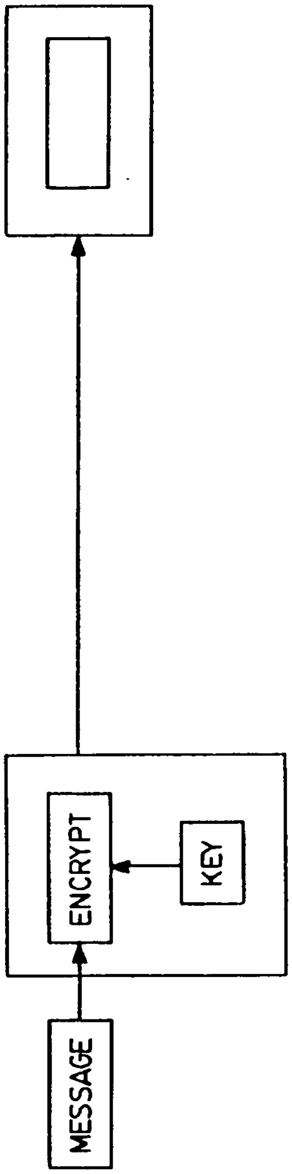
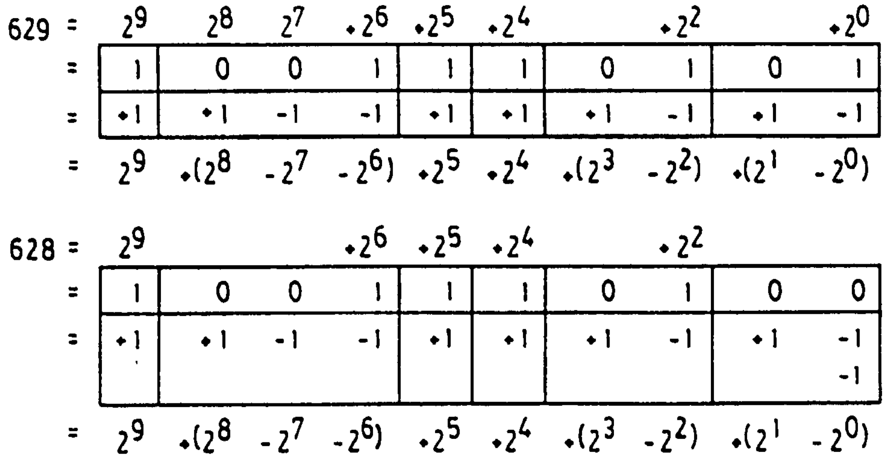
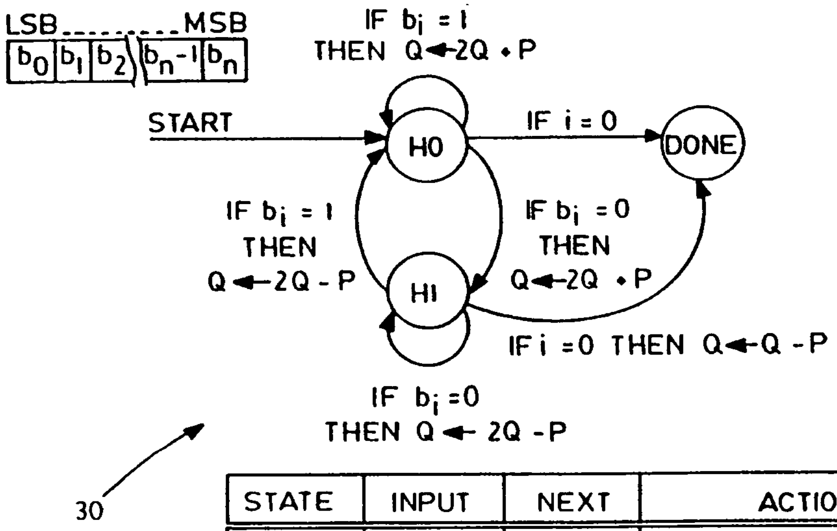


FIG. 1



629	=	2^9	2^8	2^7	$+2^6$	$+2^5$	$+2^4$		$+2^2$		$+2^0$
	=	1	0	0	1	1	1	0	1	0	1
	=	+1	+1	-1	-1	+1	+1	+1	-1	+1	-1
	=	2^9	$+(2^8$	-2^7	$-2^6)$	$+2^5$	$+2^4$	$+(2^3$	$-2^2)$	$+(2^1$	$-2^0)$
628	=	2^9			$+2^6$	$+2^5$	$+2^4$		$+2^2$		
	=	1	0	0	1	1	1	0	1	0	0
	=	+1	+1	-1	-1	+1	+1	+1	-1	+1	-1 -1
	=	2^9	$+(2^8$	-2^7	$-2^6)$	$+2^5$	$+2^4$	$+(2^3$	$-2^2)$	$+(2^1$	$-2^0)$

FIG. 2



STATE	INPUT	NEXT	ACTION
H0	$i < 0$	DONE	SUBTRACT
H0	$b_i = 0$	H1	DOUBLE, ADD
H0	$b_i = 1$	H0	DOUBLE, ADD
H1	$i < 1$	DONE	
H1	$b_i = 0$	H1	DOUBLE, SUBTRACT
H1	$b_i = 1$	H0	DOUBLE, SUBTRACT

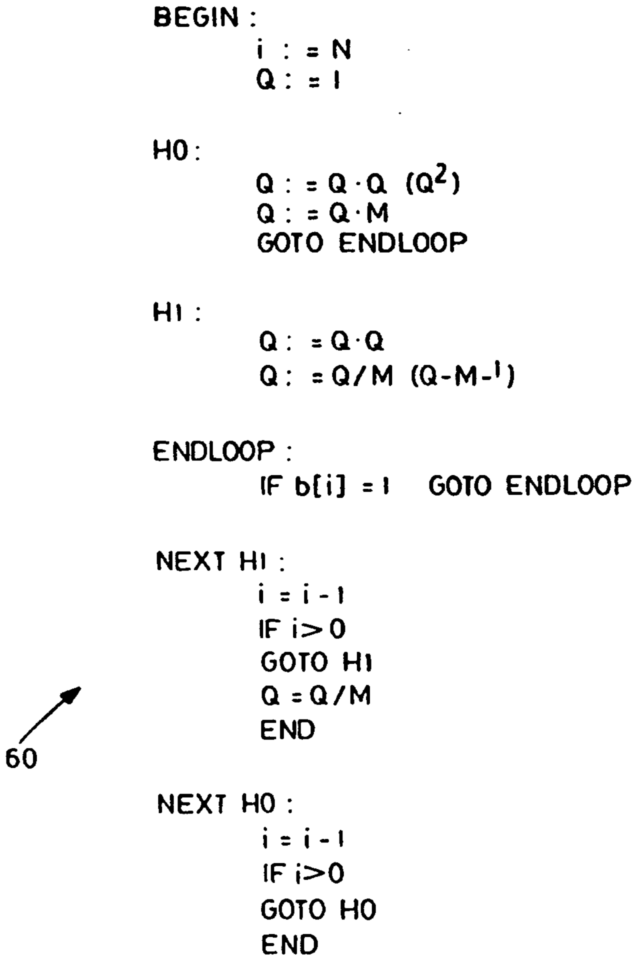
FIG. 3

```
BEGIN :
    i  := N              ; START FROM MSB                         L1
    Q  := 0              ; INITIALIZE ACCUMULATOR                 L2
    H  := 0              ; INITIALIZE STATE                       L3

LOOP :                   ; FOR ALL BITS
    Q := Q + Q           ; DOUBLE ACCUMULATOR                     L4

    IF H = 0             ; IF H STATE IS SET                      L5
      Q : = Q + P        ;   ADD BASE POINT TO ACCUMULATOR        L6
      GOTO ENDLOOP       ;                                        L7
    ELSE                 ; ELSE
      Q : = Q + (-P)     ;   SUBTRACT BASE POINT                  L8
      GOTO ENDLOOP       ;                                        L9

ENDLOOP :
    H  := b[i]           ; SET H STATE TO VALUE OF b[i]           L10
    i  := i -1           ; PROCESS NEXT BIT                       L11
    IF i>0               ; IF BIT EXISTS                          L12
      GOTO LOOP          ;   CONTINUE AT TOP OF LOOP              L13

    IF H = 0             ; IF EXISTING FROM H = 0 STATE           L14
      Q : = Q + (-P)     ;   CORRECT RESULT BY FINAL SUBTRACT     L15
    END                                                           L16
```

FIG. 4

```
BEGIN :
        i : =N               ; START FROM MSB                          LL1
        Q : =0               ; INITIALIZE ACCUMULATOR                  LL2

H0 :                         ; STATE ENTRY POINT
        Q : = Q + Q          ; DOUBLE ACCUMULATOR                      LL3
        Q : = Q + P          ; ADD BASE POINT TO ACCUMULATOR           LL4
        GOTO ENDLOOP         ; BRANCH TO END OF LOOP TESTS             LL5

H1 :                         ; STATE ENTRY POINT
        Q : = Q + Q          ; DOUBLE ACCUMULATOR                      LL6
                               SUBTRACT BASE POINT FROM
        Q : = Q + (-P)       ;                                         LL7
                               ACCUMULATOR
        GOTO ENDLOOP         ; BRANCH TO END OF LOOP TESTS             LL8

ENDLOOP :                    ; END OF LOOP TESTS
        IF b[i]=1            ; IF CURRENT BIT IS SET                   LL9
          GOTO NEXT H0       ;    FOLLOW H0 PATH                       LL10
                             ; ELSE FALL INTO H1 PATH

NEXT H1 :                    ; H1 PATH
        i : =i-1             ; PROCESS NEXT BIT                        LL11
        IF i>0               ; IF BIT EXISTS                           LL12
          GOTO H1            ;   EXECUTE H1 STATE                      LL13
        Q : = Q + (-P)       ; ELSE CORRECT RESULT AND END             LL14
        END                                                            LL15

NEXT H0 :                    ; H0 PATH
        i : =i-1             ; PROCESS NEXT BIT                        LL16
        IF i>0               ; IF BIT EXISTS                           LL17
          GOTO H0            ;   EXECUTE H0 STATE                      LL18
        END                  ; ELSE END                                LL19
```

FIG. 5



```
BEGIN :
        i : = N
        Q : = 1

H0 :
        Q : = Q·Q  ($Q^2$)
        Q : = Q·M
        GOTO ENDLOOP

H1 :
        Q :  = Q·Q
        Q :  = Q/M  (Q-M-$^1$)

ENDLOOP :
        IF b[i] = 1   GOTO ENDLOOP

NEXT H1 :
        i = i - 1
        IF i>0
        GOTO H1
        Q = Q/M
        END

NEXT H0 :
        i = i - 1
        IF i>0
        GOTO H0
        END
```

FIG. 6

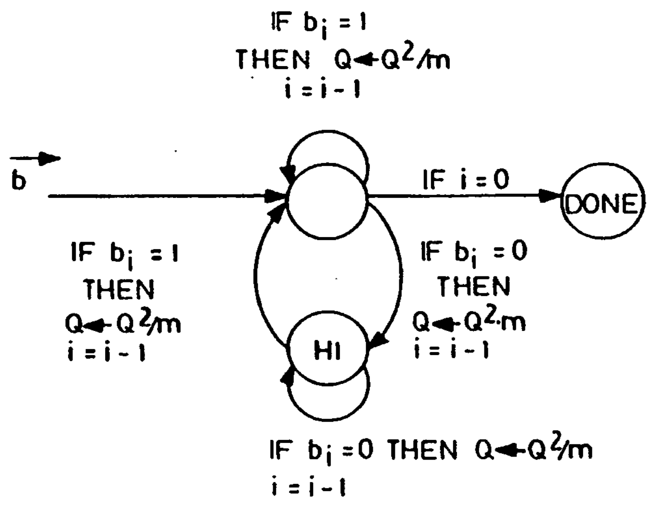

IF b_i = 1
THEN $Q \leftarrow Q^2/m$
$i = i - 1$
$\vec{b}$
IF $i = 0$
DONE
IF b_i = 1
THEN
$Q \leftarrow Q^2/m$
$i = i - 1$
IF b_i = 0
THEN
$Q \leftarrow Q^2 \cdot m$
$i = i - 1$
HI
IF b_i =0 THEN $Q \leftarrow Q^2/m$
$i = i - 1$

FIG. 7

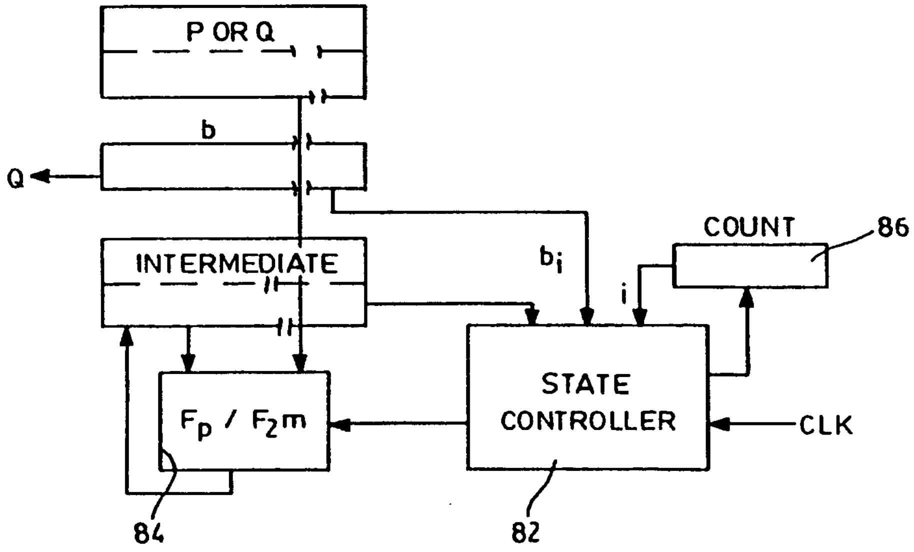

P OR Q
b
Q
INTERMEDIATE
b_i
COUNT
86
i
F_p / F_2m
STATE CONTROLLER
CLK
84
82

FIG. 8

INTERNATIONAL SEARCH REPORT

International Application No
PCT/CA 99/00658

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L9/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	KOCHER P C: "TIMING ATTACKS ON IMPLEMENTATIONS OF DIFFIE-HELLMAN, RSA, DSS, AND OTHER SYSTEMS" PROCEEDINGS OF THE ANNUAL INTERNATIONAL CRYPTOLOGY CONFERENCE (CRYPTO),DE,BERLIN, SPRINGER, vol. CONF. 16, page 104-113 XP000626590 ISBN: 3-540-61512-1 the whole document ---	1,6
A	EP 0 682 327 A (YEDA RES & DEV) 15 November 1995 (1995-11-15) abstract page 2, line 30 - line 42 page 3, line 41 - line 54 page 4, line 5 - line 26 claims 1,4; figure 3 -----	1,6

□ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search 19 November 1999	Date of mailing of the international search report 01/12/1999
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Gautier, L

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

F..../CA 99/00658

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0682327 A	15-11-1995	US 5504817 A	02-04-1996